UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/726,192	12/02/2003	Kwasi Addo Asare	RSW920030193US1 (148)	2577

46320 7590 09/02/2008

CAREY, RODRIGUEZ, GREENBERG & PAUL, LLP
STEVEN M. GREENBERG
950 PENINSULA CORPORATE CIRCLE
SUITE 3020
BOCA RATON, FL 33487

EXAMINER	
DAO, THUY CHAN	
ART UNIT	**PAPER NUMBER**
2192	

MAIL DATE	DELIVERY MODE
09/02/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 10/726,192
Filing Date: December 02, 2003
Appellants(s): ASARE ET AL.

Scott D. Paul
For Appellants

EXAMINER'S ANSWER

This is in response to the appeal brief filed June 17, 2008 appealing from the Office action mailed March 17, 2008.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The Appellants' statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The Appellants' statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

6,871,344	GRIER et al.	03-2005

(9) Grounds of Rejection

The following grounds of rejection are applicable to the appealed claims:

❑ Claims 1-2, 4-7, 11-12, and 14-17 are rejected under 35 U.S.C. 102(b) as being anticipated by Grier (US Patent No. 6,871,344).

Claim 1:

Grier discloses a machine readable storage and *a hosting environment abstraction method* (e.g., col.7: 26-33; FIG. 2A, col.7: 52 – col.8: 51) *comprising the steps of:*

enumerating each of a set of components in an application (e.g.,

col.11: 14-30, assemblies as a set of components in an application;

FIG. 7, step 712 "enumerates any dependencies ... to add dependent nodes to the dependency graph", col.21: 11-31;

FIG. 11, step 1114 also "enumerates any dependencies … so that the correct versions as specified in the configurations are bound to the application", col.23: 14-21 – emphasis added; col.7: 30-33; and

FIG. 6, block 614 "Another Assembly"? YES, looping/enumerating each assembly to identify dependencies → block 616 → FIG. 7)*;*

identifying dependencies between each component in said set (e.g.,

FIG. 7, steps 704/710/712, identify dependencies between assemblies and replace and/or add assemblies, col.20: 48-65, col.21: 1-31; and

FIG. 8, identifying assemblies N0, N2 and their dependencies with other assemblies N3, N4, …, N10, related text in col.20: 61 – col.21: 31)*; and*

organizing a generic representation of said set of components into a hierarchical structure based upon said identified dependencies (e.g., a hierarchical structure as a dependency graph 800 in FIG. 8, col.20: 48 – col.21: 10)*;*

producing a model encapsulating said hierarchical structure; and storing said model in a repository for subsequent retrieval (e.g., FIG. 2A-B, Manifests 214 and 215, col.11: 14 – col.12: 57; FIG. 3A-B, Application Manifest 204, col.16: 4-39; col.11, Table 3; FIG. 6, block 600 → block 610; col.8: 1-11)*; wherein*

said identifying step comprises the step of inspecting each component in said set for data and method member references (e.g.,

col.11: 14-41; col.21: 11-21;

FIG. 4, the step of inspecting must have been performed to "detect whether a saved activation context 302 is valid", col.18: 37-42, "map the application's requests to the proper assembly versions" col.18: 45-46; "fields", "DLL name", "pathname" following the configuration-resolution process, col.18: 48-58;

FIG. 7, steps 704/710/712, identifying and/or inspecting/evaluating dependencies using pointers, fields, publisher policy, and application policy, col.20: 48-65)

to other ones of said components in said set (e.g., col.11: 14-41; col.18: 45-62; FIG. 8, col.20: 48 – col.21: 10),

said references indicating a dependency (e.g., col.11: 14-41; col.21: 11-31; FIG. 11- Handle any new Dependencies 1114, col.22: 44 – col.23: 13), *and*

the components are application components, and the application comprises the set of components (e.g., col.11: 14-30; FIG. 7, col.20: 42 – col.21: 31).

Claim 2:

The rejection of claim 1 is incorporated. Grier also discloses *identifying dependencies between target platform resources and said components in said set; and, recording said further identified dependencies in said model* (e.g., col.8: 1-21; col.20: 48 – col.21: 10).

Claim 4:

The rejection of claim 2 is incorporated. Grier also discloses *inspecting each component in said set for data and method member references to said target platform resources* (e.g.,

FIG. 4, the step of inspecting must have been performed to "detect whether a saved activation context 302 is valid", col.18: 37-42, "map the application's requests to the

proper assembly versions" col.18: 45-46; "fields", "DLL name", "pathname" following the configuration-resolution process, col.18: 48-58;

FIG. 7, steps 704/710/712, identifying and/or inspecting/evaluating dependencies using pointers, fields, publisher policy, and application policy, col.20: 48-65, col.21: 1-31).

Claim 5:

The rejection of claim 1 is incorporated. Grier explicitly disclose *writing said hierarchical structure to a markup language document wherein tags in said markup language document demarcate individual ones of said components and said identified dependencies* (e.g., col.8: 1-21, an application manifest as an XML-formatted file; col.9 – col.12, Tables 1-3, sample assembly manifests with tags indicating individual assemblies and identified dependent assemblies).

Claim 6:

The rejection of claim 1 is incorporated. Grier also discloses *performing enumerating, identifying, organizing, producing and storing step subsequent to installing said application in a target platform* (e.g.,

FIG. 7, step 712 "enumerates any dependencies ... to add dependent nodes to the dependency graph", col.21: 11-31;

FIG. 11, step 1114 also "enumerates any dependencies … so that the correct versions as specified in the configurations are bound to the application", col.23: 14-21 – emphasis added);

FIG. 7, steps 704/710/712, identify dependencies between assemblies and replace and/or add assemblies, col.20: 48-65, col.21: 1-31).

Claim 7:

The rejection of claim 1 is incorporated. Grier also discloses *retrieving said model from said repository prior to installing a new component for use in said application* (e.g., FIG. 2A-B, Manifests 214 and 215 stored in Global Assembly Cache 212, col.11: 14 –

col.12: 57; FIG. 3A-B, Application Manifest 204 stored in Global Assembly Cache 212, col.16: 4-39).

Claims 11-12 and 14-17:

Claims 11-12 and 14-17 are machine readable storage versions, which recite the same limitations as those of claims 1-2 and 4-7, wherein all claimed limitations have been addressed and/or set forth above. Therefore, as the reference teaches all of the limitations of the above claims, it also teaches all of the limitations of claims 11-12 and 14-17.

(10) Response to Argument

Claim 1 (Brief, pp. 5-8):

a) The limitations "*enumerating each of a set of components in an application*" (claim 1, line 2 and Brief, page 5, lines 10 – page 6, line 3).

As an initial matter, examiner notes that the Appellants did not response to the ground of rejection as set forth in page 3 of the previous Office action mailed March 17, 2008. The Appellants are silent regarding the figures and text portions applied to the claim in page 3 of the Office action as follows:

Grier discloses "*enumerating each of a set of components in an application*" (e.g.,

col.11: 14-30, assemblies as a set of components in an application;

FIG. 7 and related text in col.21:11-31,

> "…Step 712 enumerates any dependencies in the assembly manifest that corresponds to this appropriate assembly, e.g., to add dependent nodes to the dependency graph. Note that if an assembly representation (node) is already in the graph for a given assembly, a pointer from the node may be added to show the dependency rather than place a new node in the dependency graph…", emphasis added;

FIG. 11 and related text in col.23: 14-21,

> "…Step 1114 enumerates any dependencies in the assembly manifest that corresponds to this appropriate assembly for handling in a similar manner. These and other identified assemblies may be handled in a similar manner so that the correct versions as specified in the configurations are bound to the application", emphasis added; and further in

FIG. 6, block 614 "Another Assembly"? YES, looping/enumerating each assembly to identify dependencies → block 616 → FIG. 7).

The Appellants further asserted, "…*'and thus as used herein a set of one or more components are also referred to as an assembly' (emphasis added). Thus, based upon the unambiguous teachings of Grier, an assembly (i.e., one assembly) is the set of components*…" (Brief, page 5, lines 18-20).

The examiner respectfully disagrees. As consistently set forth in the previous Office action, Grier teaches a set of assemblies as "*a set of components in an application*" (e.g.,

> "In general, an application manifest is an XML (extensible Markup Language) formatted file or other suitable file that comprises metadata (e.g., 206) describing an application's dependencies on shareable assembly versions, (sometimes referred to as side-by-side assemblies), and also includes metadata to describe any privatized assemblies …" (col.8: 1-6, emphasis added);
>
> "Assemblies may be dependent on other assemblies, which in turn are dependent on other assemblies, and so on. To ensure the proper versions of dependent assemblies, one or more of the assemblies (e.g., assembly 208.sub.1) each have an assembly manifest …" (col.11: 14-18, emphasis added); and
>
> "…As used herein, the term "assembly" will refer to one or more components, whether referring to a single component (e.g., one contiguous DLL) or to a plurality of components grouped together" (col.7: 30-33, emphasis added).

Accordingly, Appellants' arguments are not persuasive.

b) The limitations "*identifying dependencies between each component in said set*" (claim 1, line 3 and Brief, page 6, lines 5-10).

As an initial matter, Appellants acknowledged, "…*Instead, the Examiner's cited passage describes dependencies between assemblies*…" (Brief, page 6, lines 16-17, emphasis added).

In the Office action, "*assemblies*" in Grier (col.7: 30-33 and col.11: 14-30 as recited above) have been consistently equated with the claimed limitation "*components*" in the instant application. As acknowledged by the Appellants, the "*cited passage describes dependencies between assemblies*", which indeed describes/teaches *"identifying dependencies between each component in said set"* (e.g.,

FIG. 7, steps 704/710/712, identify dependencies between assemblies and replace and/or add assemblies, col.20: 48-65, col.21: 1-31*);* and

FIG. 8, identifying assemblies N0, N2 and their dependencies with other assemblies N3, N4, …, N10, related text in col.20: 61 – col.21: 31).

Accordingly, Appellants' arguments are not persuasive.

c) The limitations "*producing a model encapsulating said hierarchical structure; and storing said model in a repository for subsequent retrieval*" (claim 1, lines 6-7 and Brief, page 6, line 22 to page 7, line 12).

As an initial matter, in the originally filed disclosure, the Appellants defined "…Both can be recorded in a dependency model 170 of the application, for example within an XML formatted document." (specification, page 10, [0021], lines 9-10, emphasis added).

The examine also notes that the Appellants did not response to the ground of rejection as set forth in page 4 of the previous Office action mailed March 17, 2008. The Appellants are silent regarding the figure and text portions applied to the claim in page 4 of the Office action as follows:

"*organizing a generic representation of said set of components into a hierarchical structure based upon said identified dependencies"* (e.g., a hierarchical structure as a dependency graph 800 in FIG. 8, col.20: 48 – col.21: 10).

The ground of rejections clearly equated the claimed limitation "*a hierarchical structure*" with "*a dependency graph 800 in FIG.8*" and later, the claimed limitation "*a model encapsulating said hierarchical structure*" with "*application/assembly manifest*" as

a model encapsulating said "*dependency graph 800*" (claimed limitation "hierarchical structure") as follows:

"producing a model encapsulating said hierarchical structure; and storing said model in a repository for subsequent retrieval" (e.g., FIG. 2A-B, Manifests 214 and 215, col.11: 14 – col.12: 57; FIG. 3A-B, Application Manifest 204, col.16: 4-39,

col.11 illustrates a sample of manifest in an XML formatted document, which is equivalent to Dependency Model 170 (FIG. 1) of Appellants as noted above:

data member references

method member references (method calls to other assemblies)

Each assembly manifest describes the assembly and includes information about its individual assemblies, including, for example, the name and version of the assembly, the items (program files, resources) that make up the assembly, and the binding path to items within the assembly (e.g., for Win32 DLLs this is the location of the DLL relative to the root of the assembly, whereas for COM Servers this is the CLSID (class identifier), ProgID (programmatic identifier) and other COM metadata). The assembly manifest may also include any dependencies on other assemblies, object classes and global names.

A sample assembly manifest is set forth in TABLE 3 below:

TABLE 3

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1"
manifestVersion="1.0">
    <assemblyIdentity type="win32"
name="Microsoft.Tools.VisualCPlusPlus.Runtime-Libraries"
version="6.0.0.0" processorArchitecture="x86"
publicKeyToken="6595b64144ccf1df"/>
    <file name="mfc42u.dll"
hash="3eab067f82304bf271ed38112a4ccdf46094e8a"
hashalg="SHA1">
        <comClass description="Font Property Page"
clsid="{08E35200-8F91-11CE-9DE3-00AA004BB851}"/>
        <comClass description="Color Property Page"
clsid="{08E35201-8F91-11CE-9DE3-00AA004BB851}"/>
        <comClass description="Picture Property Page"
clsid="{08E35202-8F91-11CE-9DE3-00AA004BB851}"/>
    </file>
    <file name="mfc42.dll"
hash="ac72753e5bb26446d88a48e8f0aaae769a962338"
hashalg="SHA1"/>
    <file name="a1.dll"
hash="a7312a1f6cfb864330f1e0540458de60a6ed5ec5"
hashalg="SHA1">
        <comClass description="Registrar Class"
clsid="{44ECB53A-400F-11D0-9DCD-00A0C90391D3}"
```

FIG. 6, block 600 "Any Application Manifest" → "YES" → block 610 "Build Partial-Dependency Graph from Application Manifest", emphasis added, i.e., application manifest (an XML formatted document/a model) encapsulating a dependency graph (a hierarchical structure);

and further in col.8: 1-11,

> "In general, an application manifest is an XML (extensible Markup Language) formatted file or other suitable file that comprises metadata (e.g., 206) describing an application's dependencies on shareable assembly versions, (sometimes referred to as side-by-side assemblies), and also includes metadata to describe any privatized assemblies (described below). For example, the application manifest 204 specifies in its dependency data 206 a dependency on a particular version (e.g., v1.0.0.0) of a shared assembly, assembly.sub.x 208.sub.1, as represented in FIG. 2A by the arrow between blocks 206 and 208.sub.1. Note that the application manifest 204 may also specify dependencies on other assemblies …", emphasis added).

Accordingly, Appellants' arguments are not persuasive.

d) The limitations "*said identifying step comprises the step of inspecting each component in said set for data and method member references to other ones of said components in said set, said references indicating a dependency*" (claim 1, lines 8-10 and Brief, page 7, line 15 to page 8, line 16).

As set forth in the previous Office action, page 4, Grier explicitly teaches:

> col.11, 14-30, "Assemblies may be dependent on other assemblies, which in turn are dependent on other assemblies, and so on. To ensure the proper versions of dependent assemblies, one or more of the assemblies (e.g., assembly 208.sub.1) each have an

assembly manifest (e.g., 214.sub.1) associated therewith that specifies its corresponding assembly's dependencies …" (i.e., inspecting each assembly for data member references such as version data "*…to ensure the proper versions of dependent assemblies… that specifies its corresponding assembly's dependencies*", emphasis added);

col.11: 31-39, "Each assembly manifest describes the assembly and includes information about its individual assemblies, including, for example, the name and version of the assembly, the items (program files, resources) that make up the assembly, and the binding path to items within the assembly (e.g., for Win32 DLLs this is the location of the DLL relative to the root of the assembly, whereas for COM Servers this is the CLSID (class identifier), ProgID (programmatic identifier) and other COM metadata)" (i.e., inspecting each assembly for data member references such as name, version, binding path, class identifier CLSID, programmatic identifier ProgID, COM metadata, emphasis added);

col.11: 40-41, "The assembly manifest may also include any dependencies on other assemblies, object classes and global names" (i.e., assemblies as Dynamic Link Libraries DLLs (col.7: 30-33), inspecting each assembly/library for method call references, parent assembly/library (as a caller) calls/depends on a child assembly/library (as a callee), emphasis added);

col.21: 11-21, "At this time, the appropriate assembly version is known, as specified in the manifest and as altered via any configuration instructions, as described above. Step 712 enumerates any dependencies in the assembly manifest that corresponds to this appropriate assembly, e.g., to add dependent

> nodes to the dependency graph. Note that if an assembly representation (node) is already in the graph for a given assembly, a pointer from the node may be added to show the dependency rather than place a new node in the dependency graph" (i.e., inspecting each assembly for member pointer references, using pointers as a member of the parent node/assembly pointing to the child node/assembly to indicate dependencies, emphasis added).

Accordingly, Grier fully teaches "*said identifying step comprises the step of inspecting each component in said set for data and method member references to other ones of said components in said set, said references indicating a dependency*" as claimed.

In conclusion, Appellants' arguments regarding claim 1 are not persuasive.

Dependent Claim 2 (Brief, pp. 8-9):

Grier also discloses *"identifying dependencies between target platform resources and said components in said set; and, recording said further identified dependencies in said model"* (wherein col.8: 1-21 discloses "privatized assemblies" (col.8: 6), which distinguish with shared assemblies (col.8: 32-36) as different target platform resources; and wherein col.20: 48 – col.21: 10 discloses a plurality of "publisher configuration" (col.20: 48-52) as different publishers/vendors, i.e., target platform resources).

Accordingly, Appellants' arguments are not persuasive.

Dependent Claim 4 (Brief, page 9):

Grier also discloses *inspecting each component in said set for data and method member references to said target platform resources* (e.g.,

FIG. 4, the step of inspecting must have been performed to "detect whether a saved activation context 302 is valid", col.18: 37-42, "map the application's requests to the

proper assembly versions" col.18: 45-46; "fields", "DLL name", "pathname" following the configuration-resolution process, col.18: 48-58, i.e., data and method member references as "fields", "DLL name", "pathname", caller/callee);

FIG. 7, steps 704/710/712, identifying and/or inspecting/evaluating dependencies using pointers, fields, publisher policy, and application policy, col.20: 48-65, col.21: 1-31. i.e., policies and different publishers/vendors with different target platform resources).

Accordingly, Appellants' arguments are not persuasive.

Dependent Claim 5 (Brief, page 9):

Appellants asserted, "...already noted above with regard to claim 1, an application manifest does not correspond to the claimed hierarchical structure. Thus, the Examiner's cited passages fails to establish that Grier identically discloses the limitations at issue in claim 5 within the meaning of 35 U.S.C. § 102".

Again, the examine notes that the Appellants did not response to the ground of rejection as set forth in page 4 of the previous Office action mailed March 17, 2008. The ground of rejections clearly equated the claimed limitation "a hierarchical structure" with "a dependency graph 800 in FIG.8" and the claimed limitation "a model encapsulating said hierarchical structure" as Manifest 214, 215 as well as application/assembly manifest.

Grier also discloses *writing said hierarchical structure to a markup language document wherein tags in said markup language document demarcate individual ones of said components and said identified dependencies* (e.g., col.8: 1-21, an application manifest as an XML-formatted file; col.9 – col.12, Tables 1-3, sample assembly manifests with tags indicating individual assemblies and identified dependent assemblies).

Accordingly, Appellants' arguments are not persuasive.

Dependent Claim 6 (Brief, pp. 9-10):

The rejection of claim 1 is incorporated. Grier also discloses *performing enumerating, identifying, organizing, producing and storing step subsequent to installing said application in a target platform* (e.g.,

FIG. 7, step 712 "enumerates any dependencies ... to add dependent nodes to the dependency graph", col.21: 11-31;

FIG. 11, step 1114 also "enumerates any dependencies … so that the correct versions as specified in the configurations are bound to the application", col.23: 14-21 – emphasis added);

FIG. 5, Runtime Version-Matching Mechanism 500, col.19: 22-50;

FIG. 7, steps 704/710/712, identify dependencies between assemblies and replace and/or add assemblies, col.20: 48-65, col.21: 1-31;

wherein said steps subsequent to installing said application in a target platform as illustrated in FIG. 2A, Application 200 installed in Application Folder 202).

Accordingly, Appellants' arguments are not persuasive.

Dependent Claim 7 (Brief, pp. 10):

Grier also discloses *retrieving said model from said repository prior to installing a new component for use in said application* (e.g.,

FIG. 2A-B, Manifests 214 and 215 stored in Global Assembly Cache 212, col.11: 14 – col.12: 57, and Publisher Configuration installs new component;

FIG. 3A-B, Application Manifest 204 stored in Global Assembly Cache 212 and retrieved for update as in FIG. 7, block 710 and 712, col.16: 4-39).

Accordingly, Appellants' arguments are not persuasive.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejection should be sustained.

Respectfully submitted,

/Thuy Dao/
Examiner, Art Unit 2192

Conferees:

/Tuan Q. Dam/
Tuan Q. Dam
Supervisory Patent Examiner, Art Unit 2192

/Wei Y Zhen/
Supervisory Patent Examiner, Art Unit 2191
Wei Y. Zhen
Supervisory Patent Examiner, Art Unit 2191